Assignment 4

Student ID: 120090645   Name: Haopeng Chen

Extra Credit

Extra work for `support boundary split` and `support mesh with boundary` is finished, and will be elaborated in the following.

Details

Task 1

The *de Casteljau's Alogirithm* is given m points with identical dimensions (i.e., `Vector2D` or `Vector3D`) and a ratio t, using the `lerp()` function (i.e. $p_{\text{new}} = lerp(p_i, p_{i+1}, t) = (1-t)p_i + tp_{i+1}$), to convert them to $m-1$ points. And repeat this step until the result is one point.

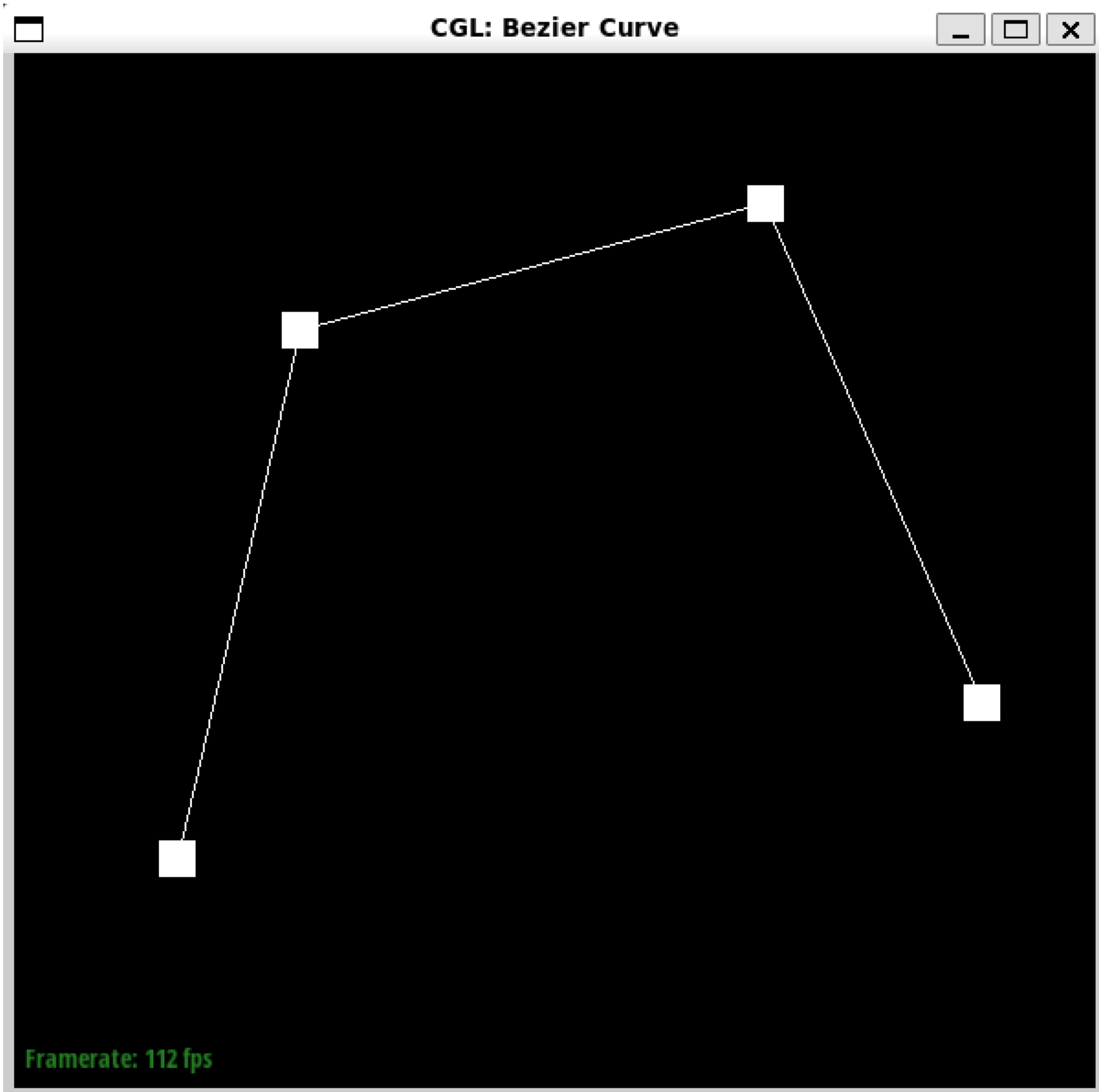
Implement

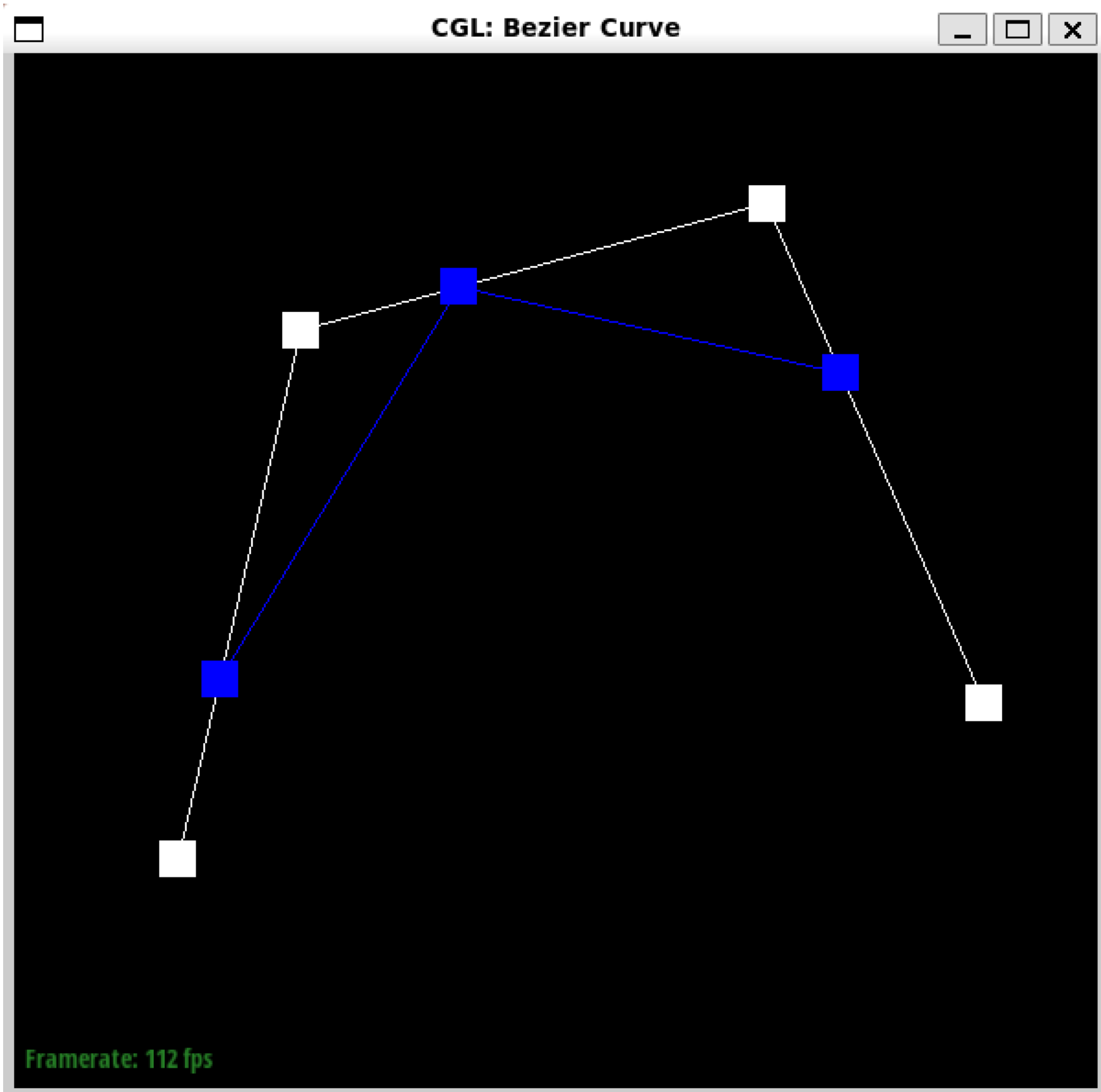
First traverse the input, applying the `lerp()` function on every two neighboring points and push the result to a `vector`. Finally return this `vector`. Do this recursively to get one point.

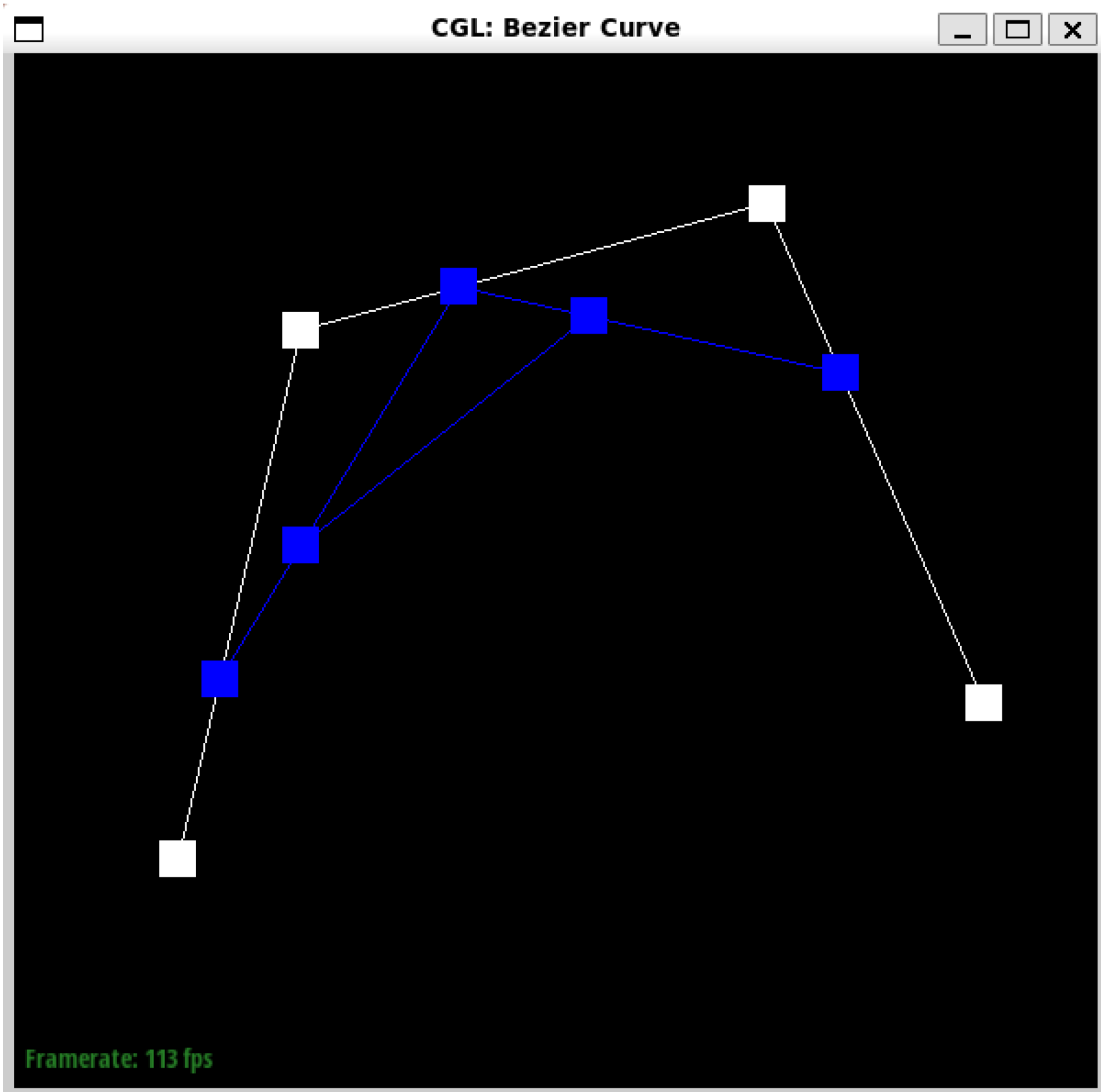
lerp() implementation in 2D

```
Vector2D lerp(const Vector2D& point1, const Vector2D& point2, double ratio)
    {
        return (1 - ratio) * point1 + ratio * point2;
    }
```

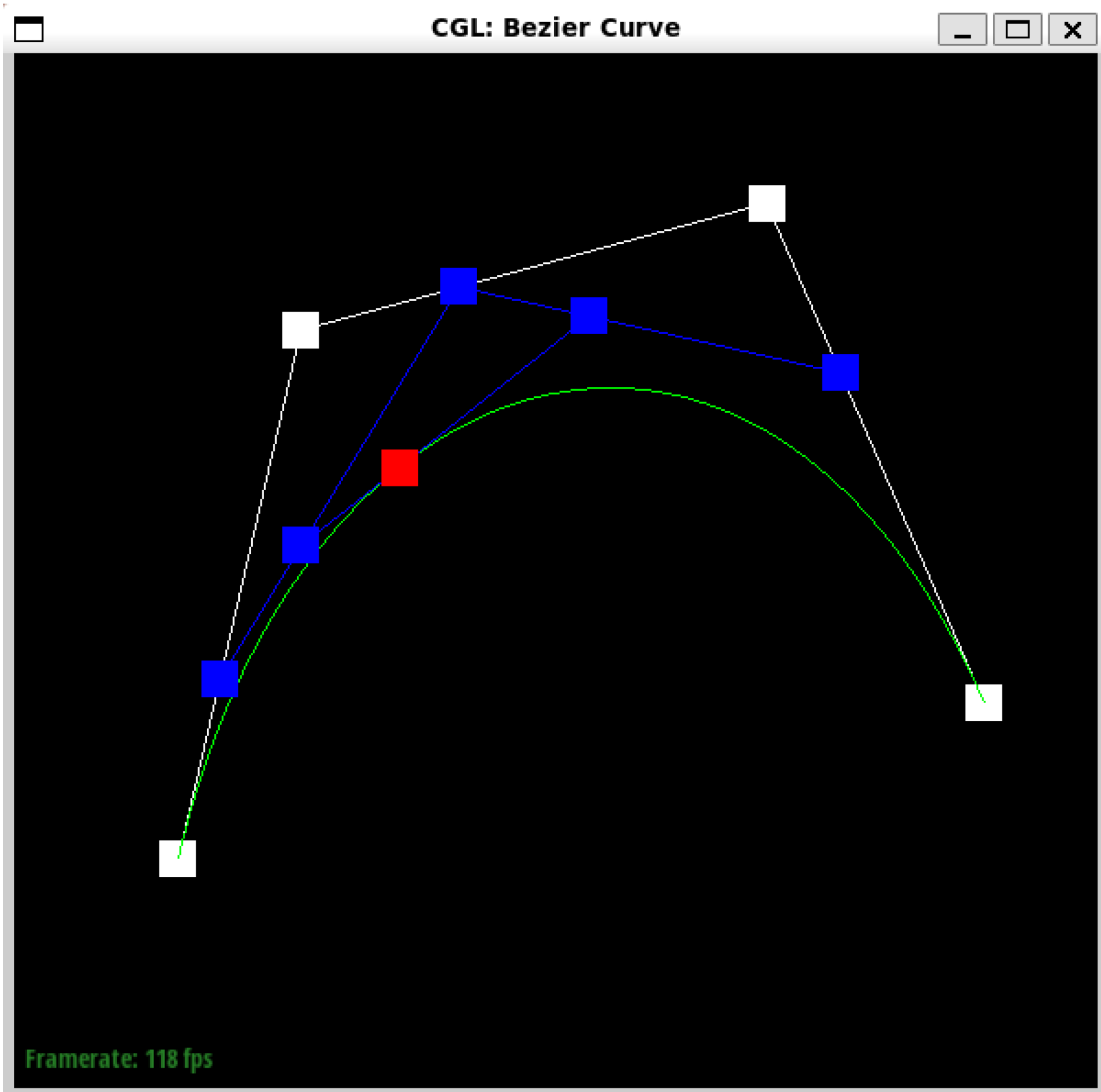
Result

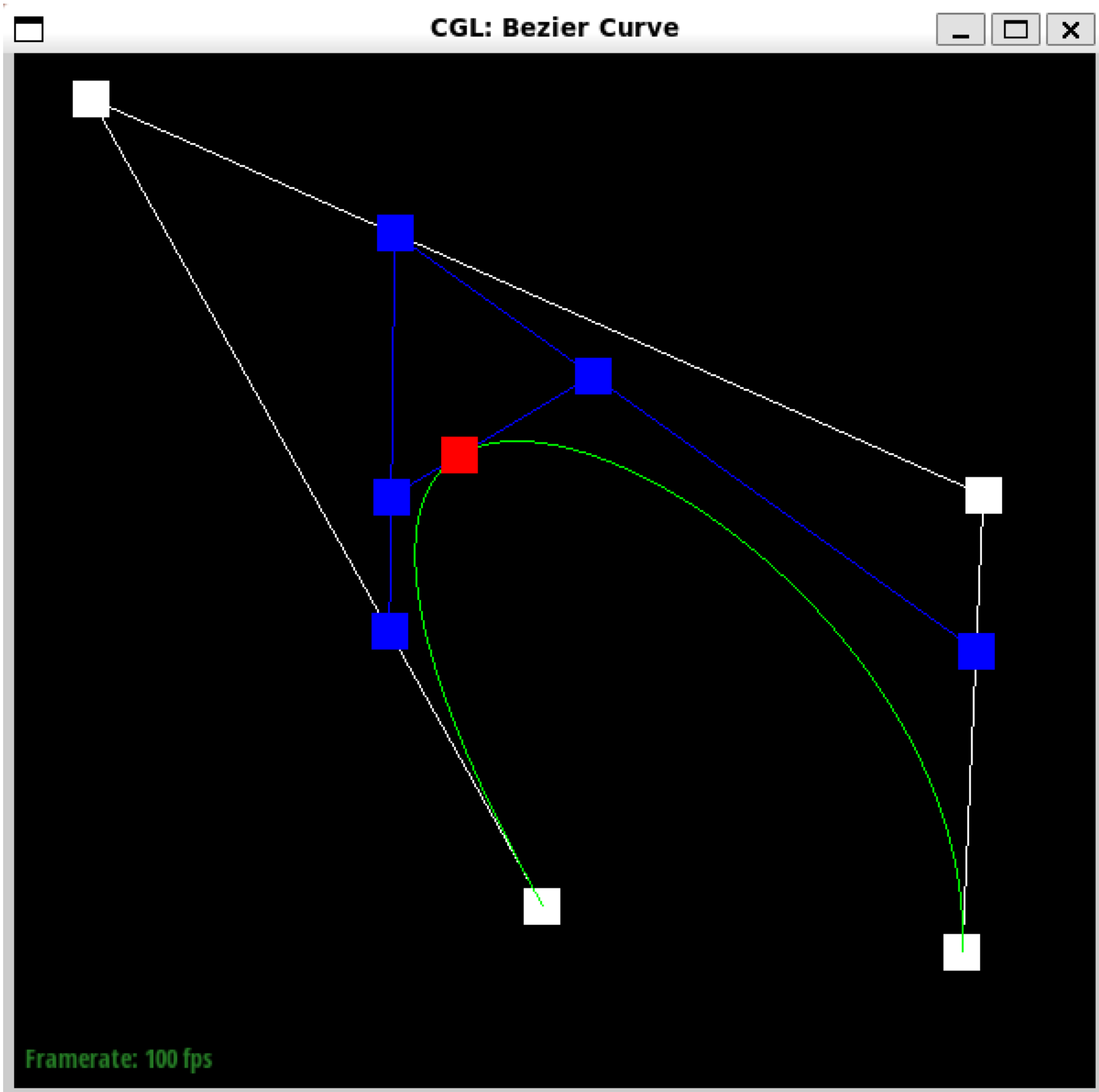

CGL: Bezier Curve
Framerate: 112 fps


CGL: Bezier Curve
Framerate: 112 fps


CGL: Bezier Curve
Framerate: 113 fps

![image-20230402140344024](./Assignment%204.assets/image-20230402140344024.png





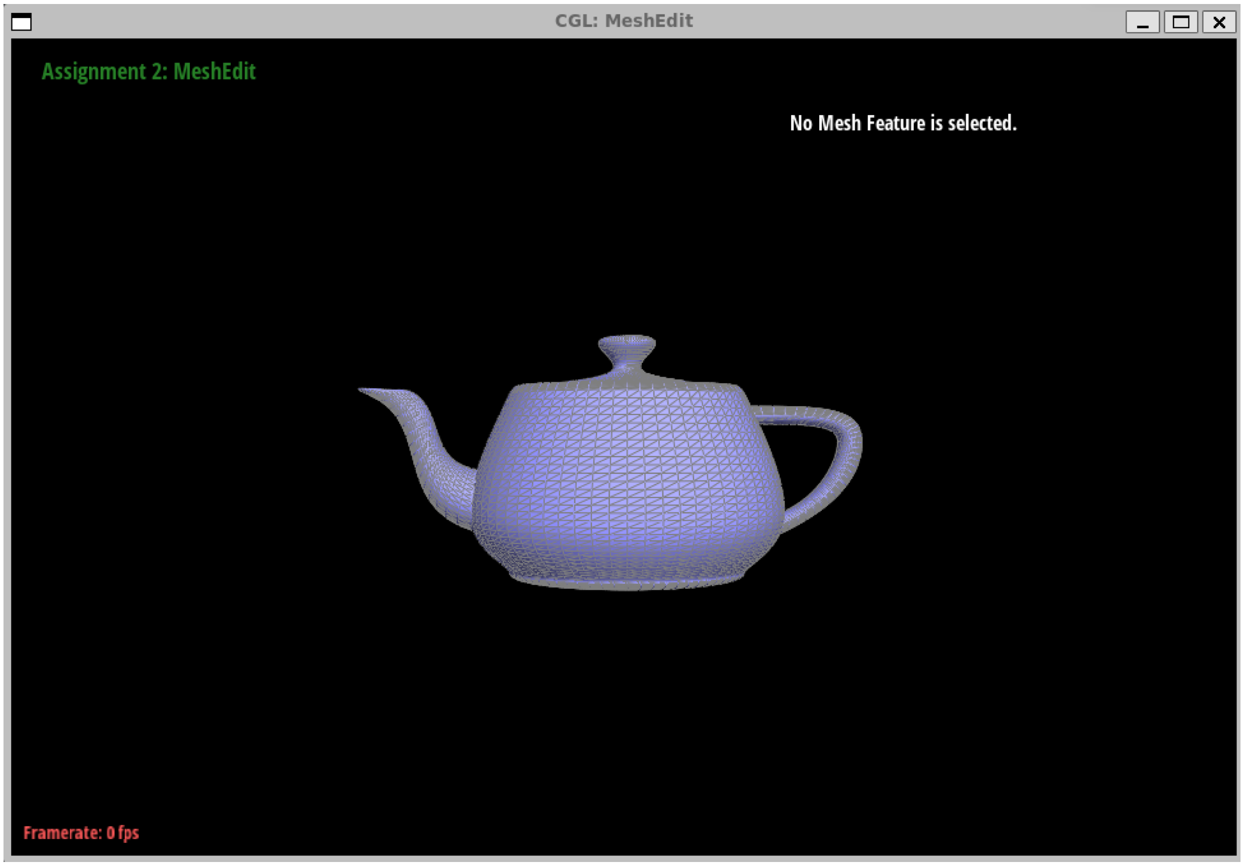
Task 2

Through *de Casteljau Algorithm*, we can get a Bezier point finally. Bezier surface can be seen as a list of Bezier curves with identical points. By evaluate horizontal Bezier curves separately and collect the outcome we get a vertical Bezier curve, then evaluate this curve by *de Casteljau Algorithm* we can get the Bezier surface.

lerp() implementation in 3D

```
Vector3D lerp(const Vector3D& point1, const Vector3D& point2, double ratio)
{
    return (1 - ratio) * point1 + ratio * point2;
}
```

Result

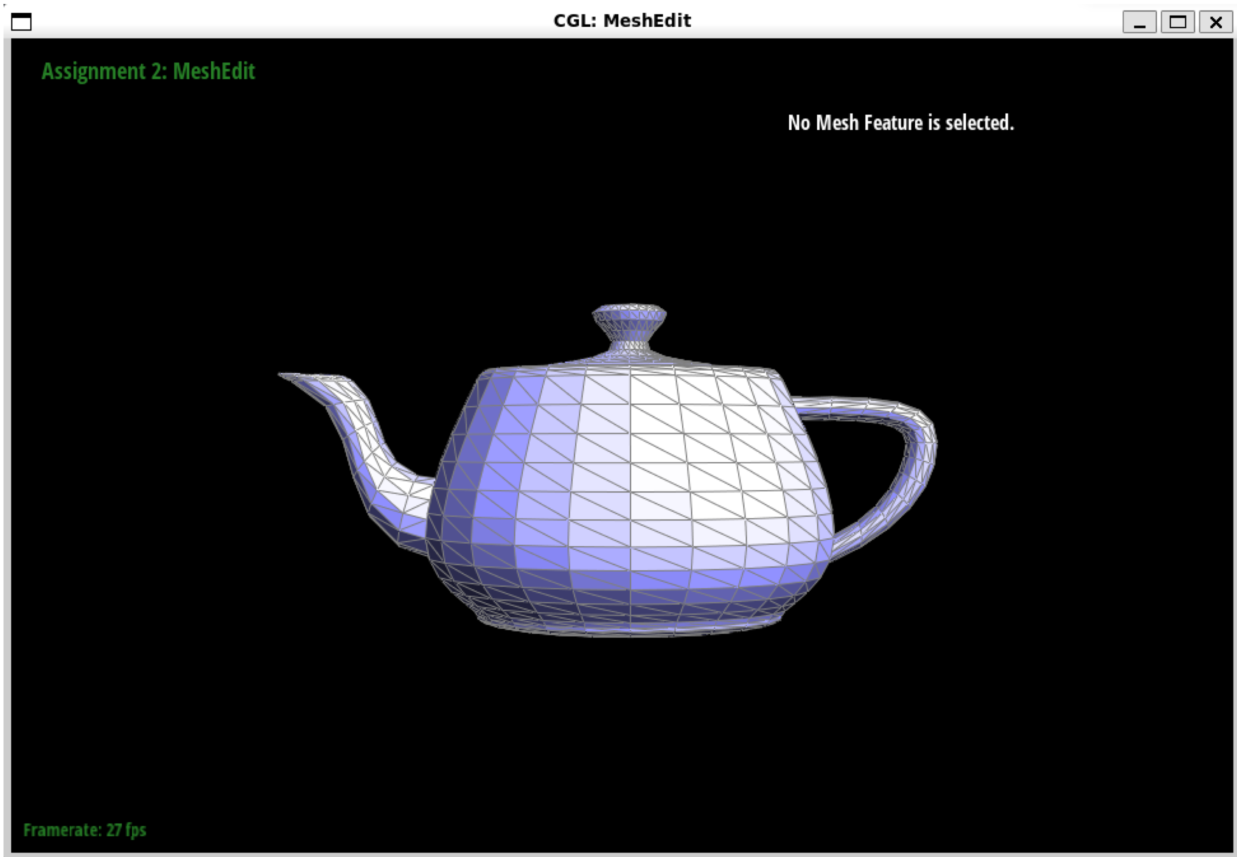


Task 3

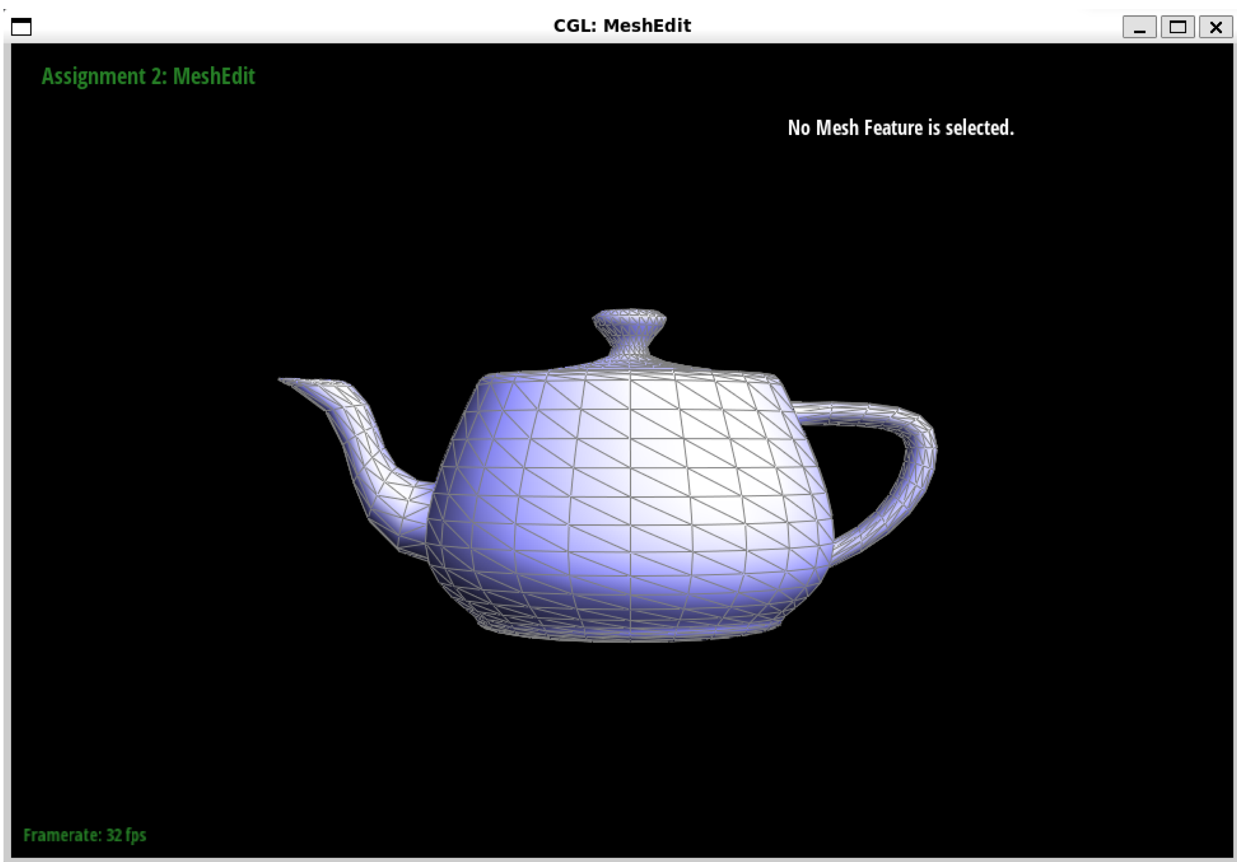
The vertex normal is calculated by area weighted surrounding faces' normal. The face normal is provided. The face area can be calculated by `0.5 * cross(a, b).norm()`, add the weighted face normal to result and normalize the result, we get the vertex normal.

Result

flat shading



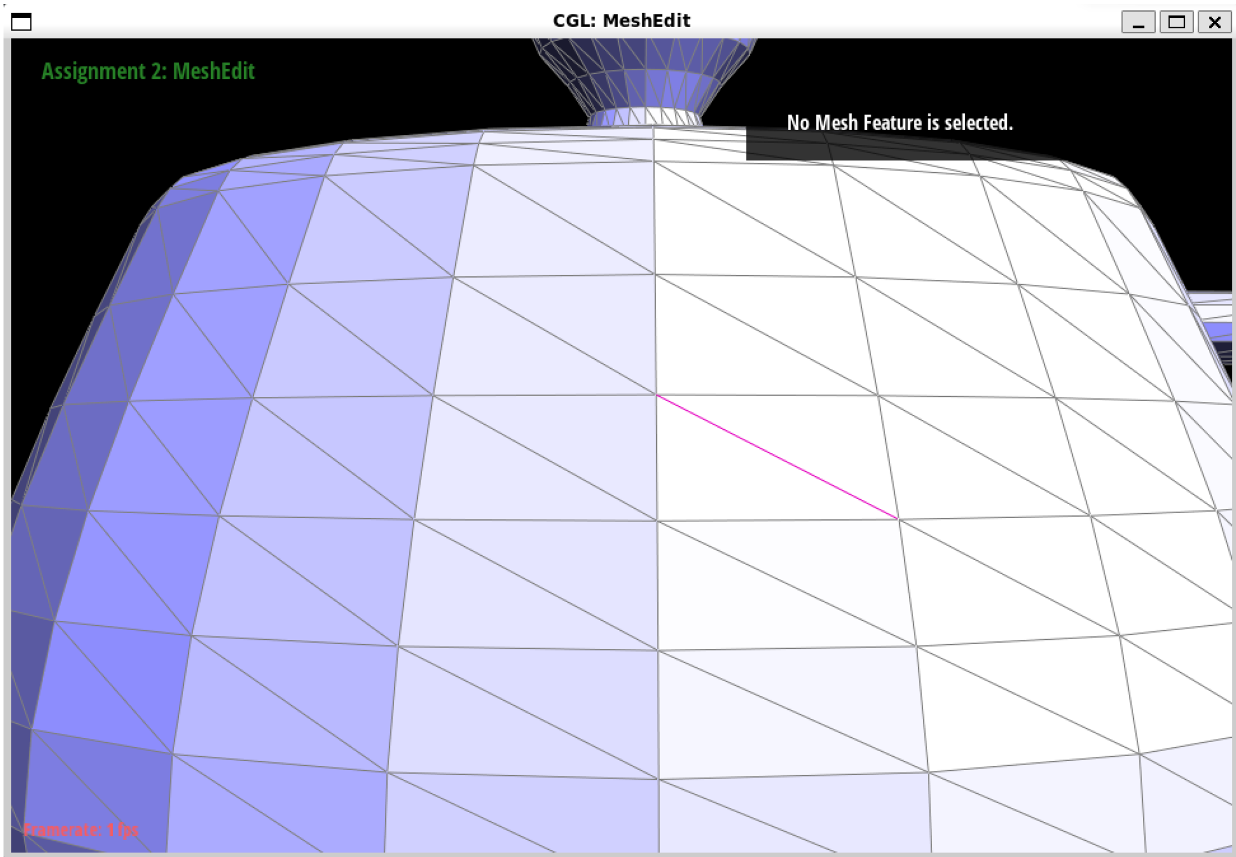
Phong shading

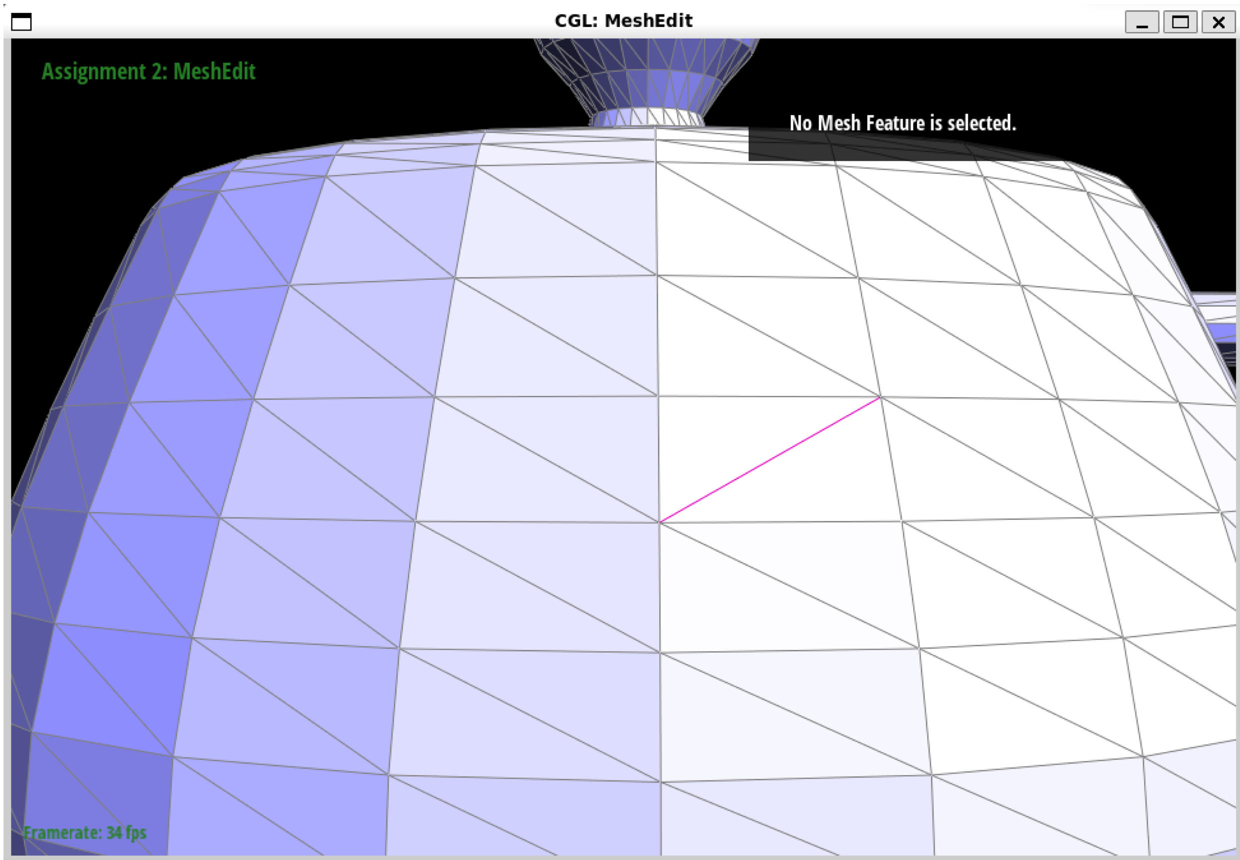


Task 4

If the selected edge is on the boundary, do nothing. Otherwise, we should access the twin and all relevant vertices and edges, and map them into corresponding members.

Result



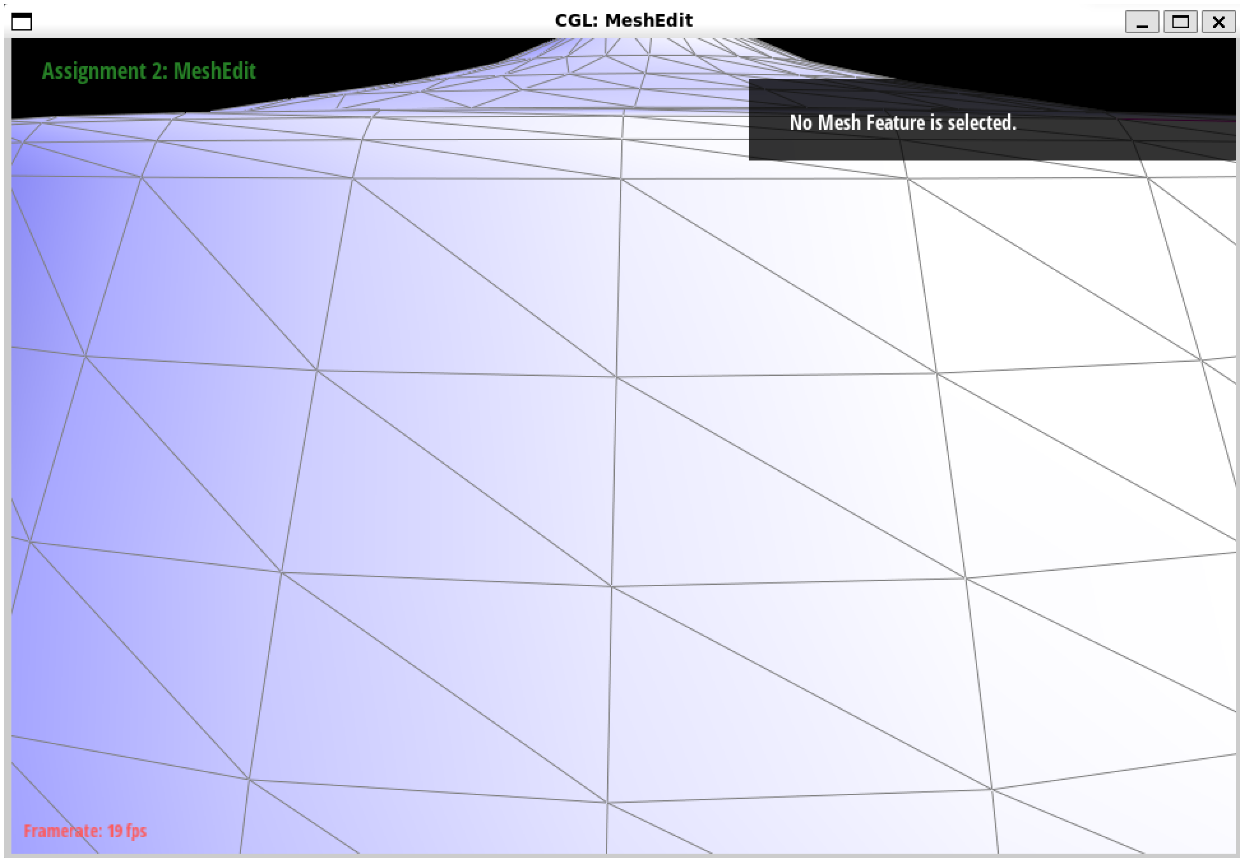


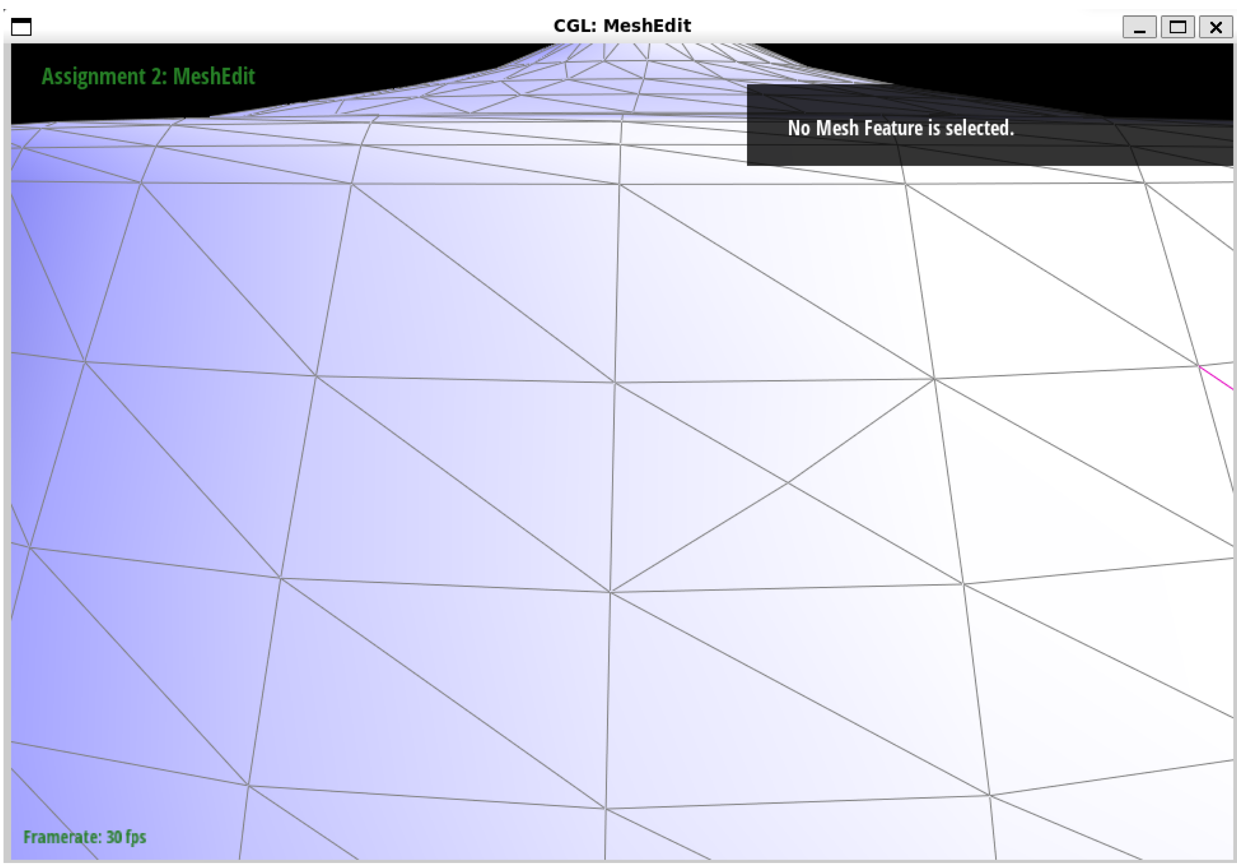
Task 5

For interior, First we get the old mesh elements may need to change: `e0, v0,v1,v2,v3, f1,f2,h1,h2,h1_1,h1_2,h2_1,h2_2` , then create new mesh elements `e2-e4, f3-f4,h3-h8,m` using given new() function. If `e0->isNew` , we just set the position of `m` to `e0->newPosition` , if not, set the position of m to midpoint of `e0`. Then based on the above diagram, set the relationship of mesh elements.

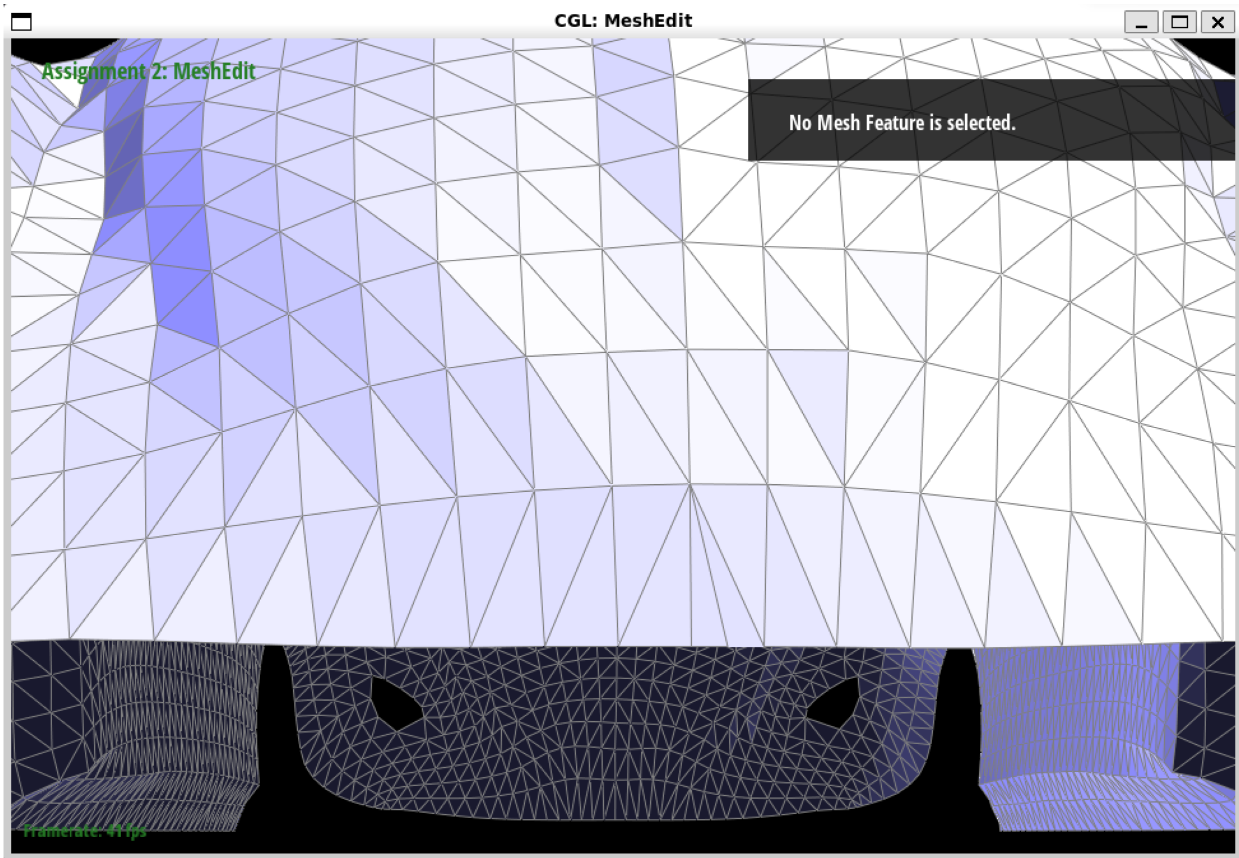
Result

inferior split





boundary split

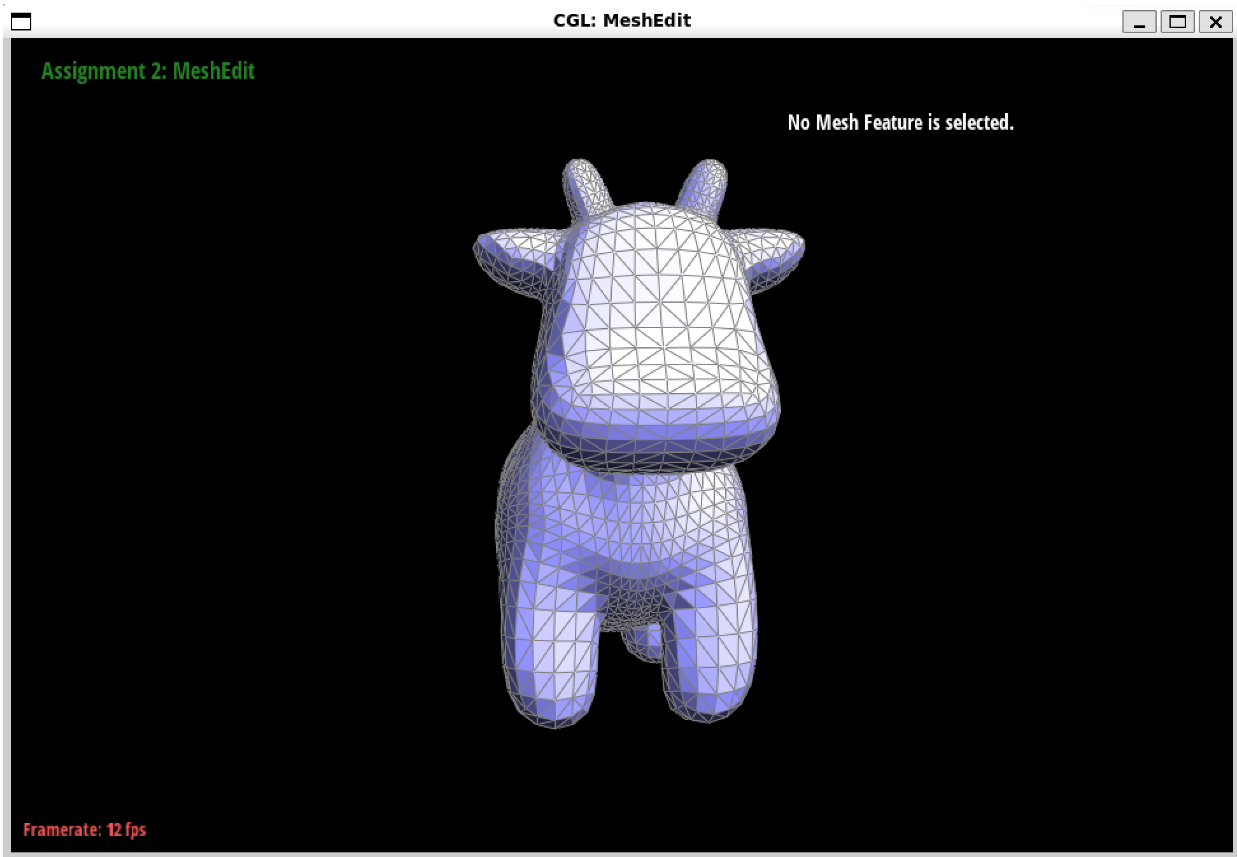


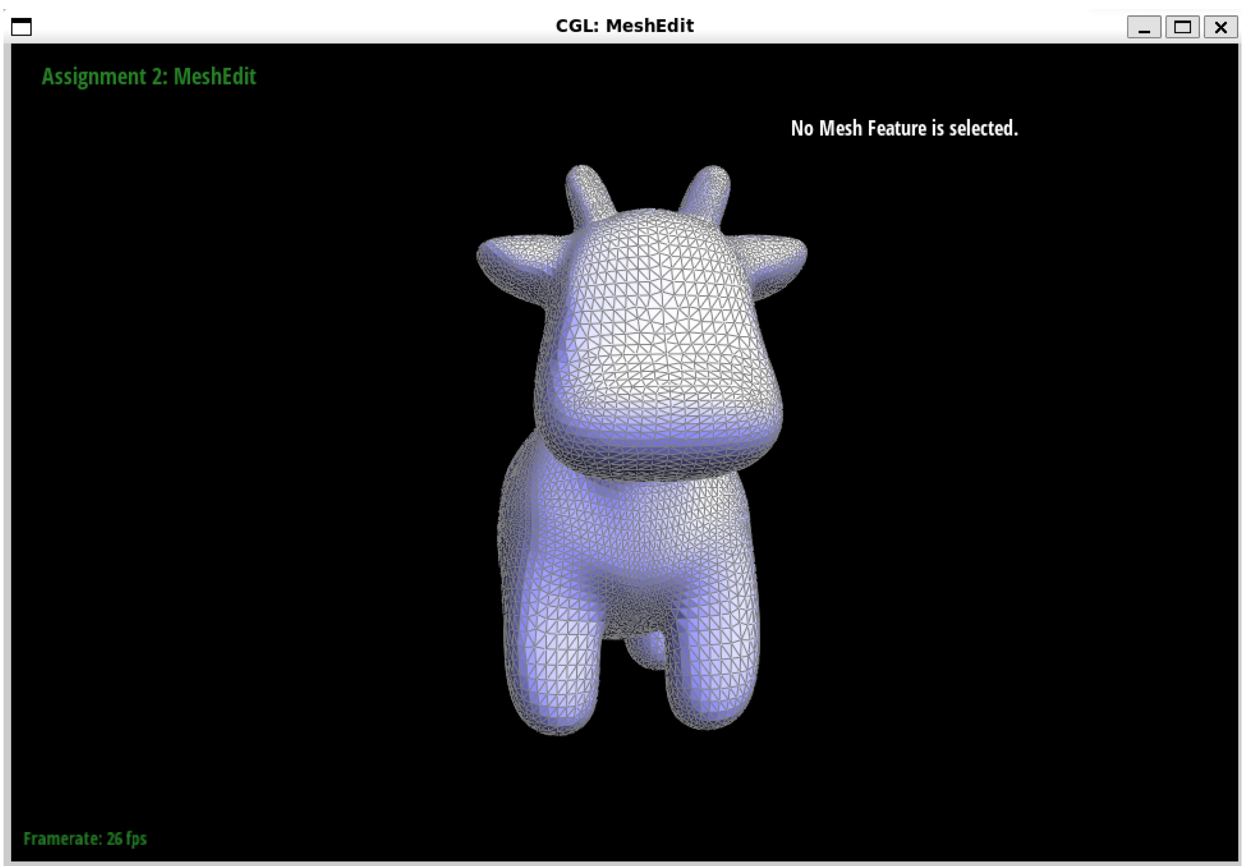
Task 6

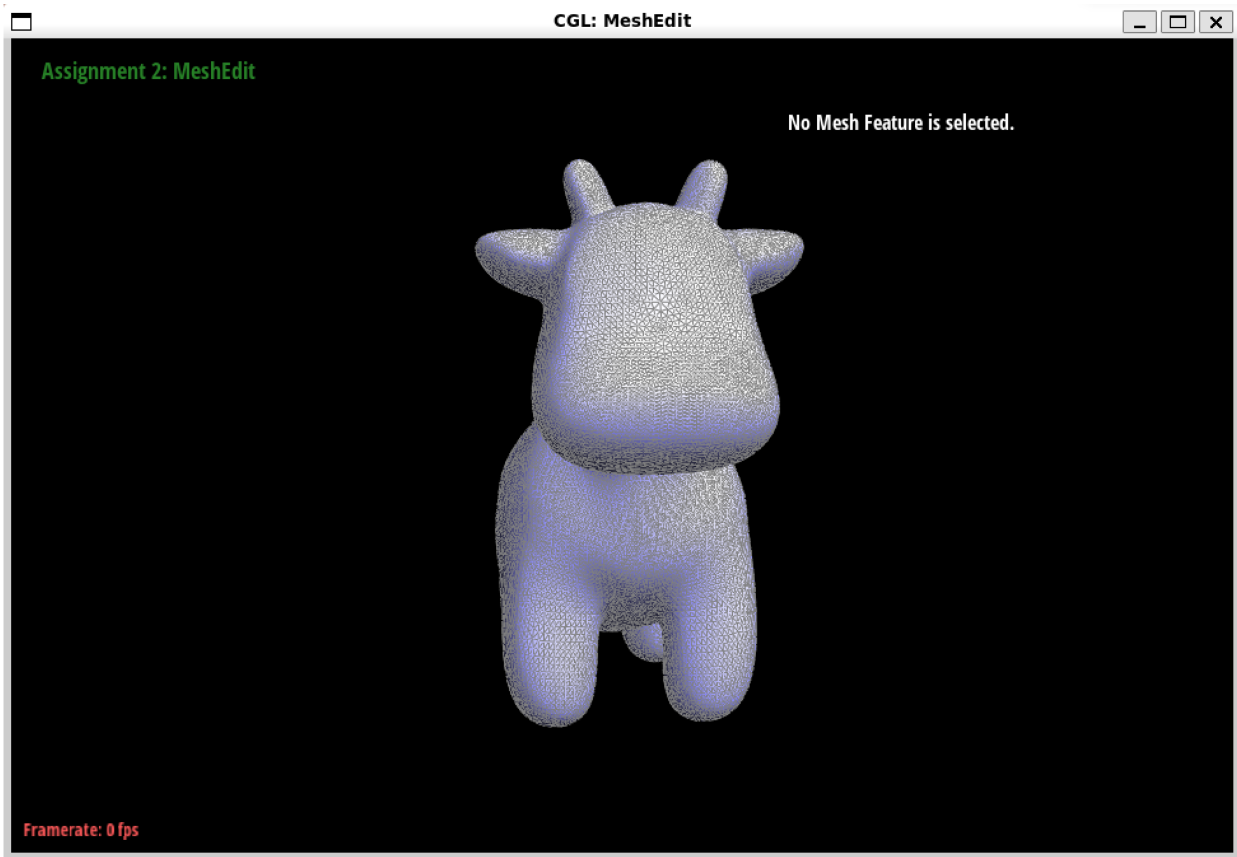
To implement this task: First we calculate the new position for old vertices. `v->newPosition=(1-n*u)*v->position+u*sum_surrounding` , `sum_surrounding` is the sum of positions of neighboring points position. Then we calculate the position of new points by `e>newPostion=0.125*(sum_v_neighbor_position)+0.375*(sum_v_on_position)` and store the position to `e->newPosition`. Then we call the `splitEdge()` function and set the value of new vertices, half_edges, edges, faces. Then for each new created edge, if it connects one old vertex and one new vertex, `flip()` it. Finally we set the old vertices position to new position.

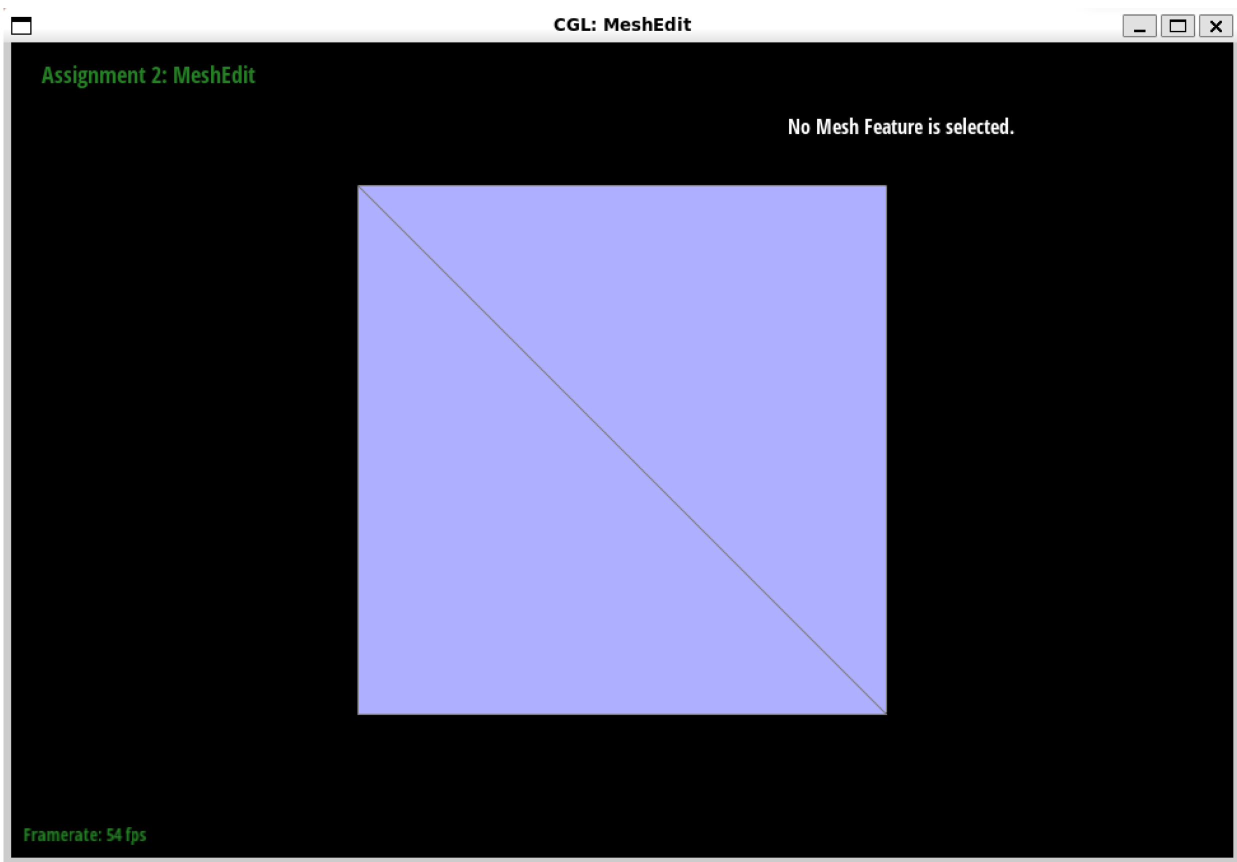
For mesh with boundary, the new position of boundary old vertices is `v->newPosition=0.75*v-position+0.125*sum_two_surrounding`, `sum_two_surrounding` is the sum of position of neighbor boundary vertices. the position of new boundary vertex is `e->newPosition=0.5*(v1->position+v2-position)`, v1, v2 are the vertices of edge.

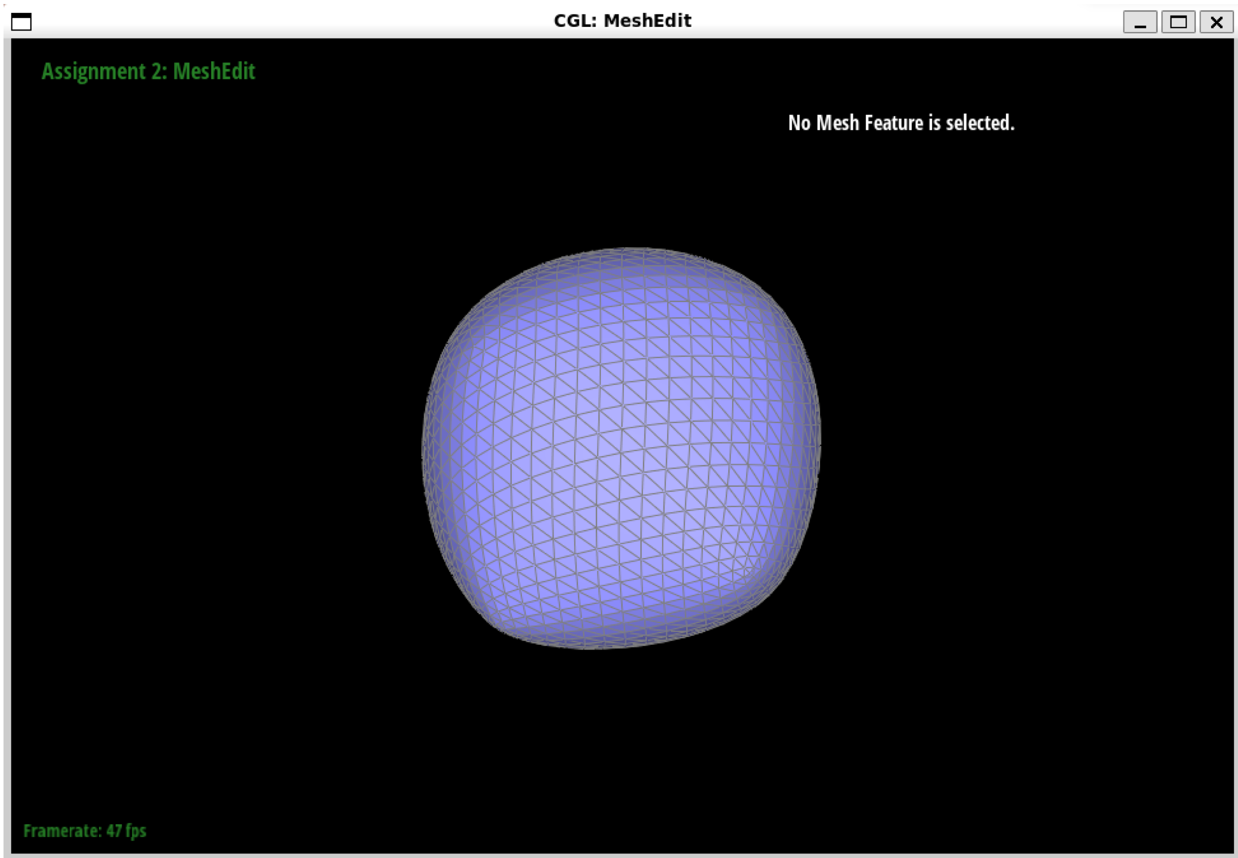
Result

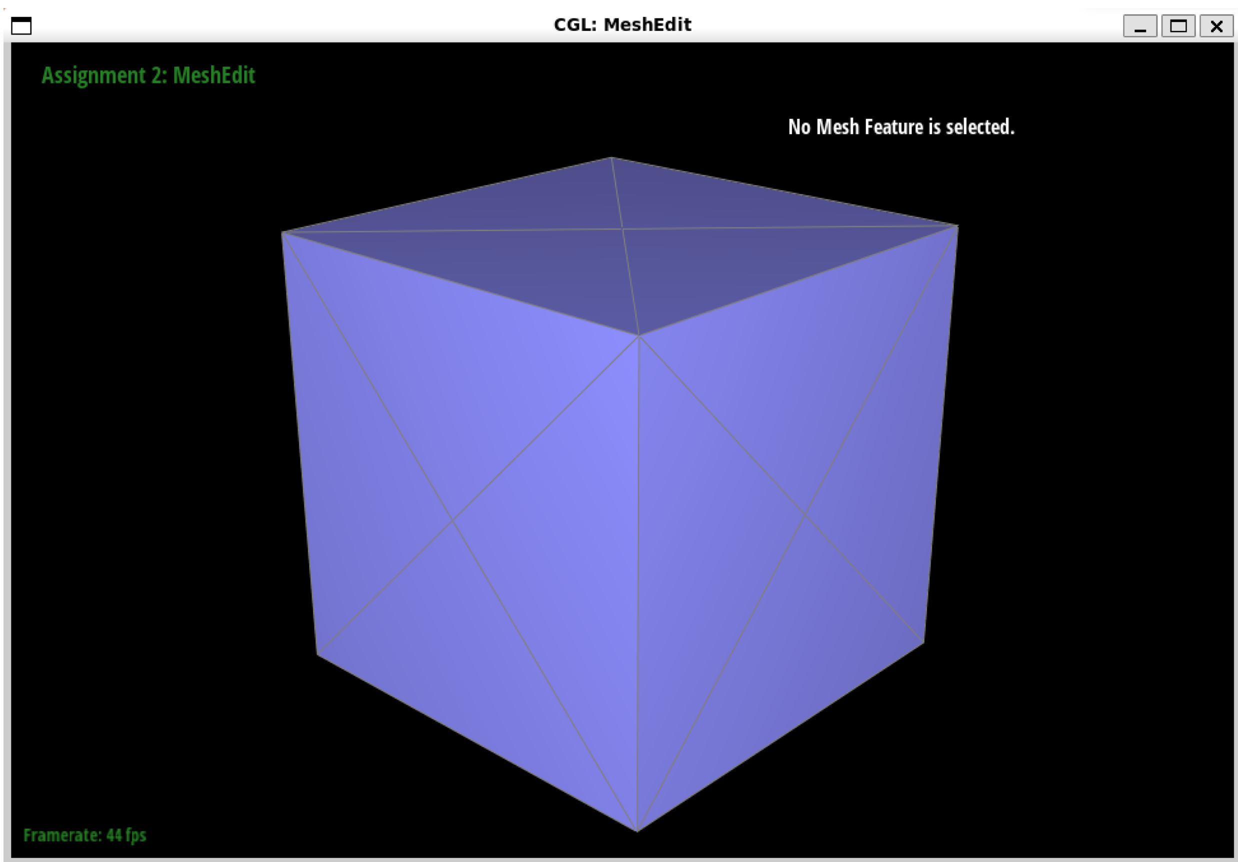

CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 12 fps

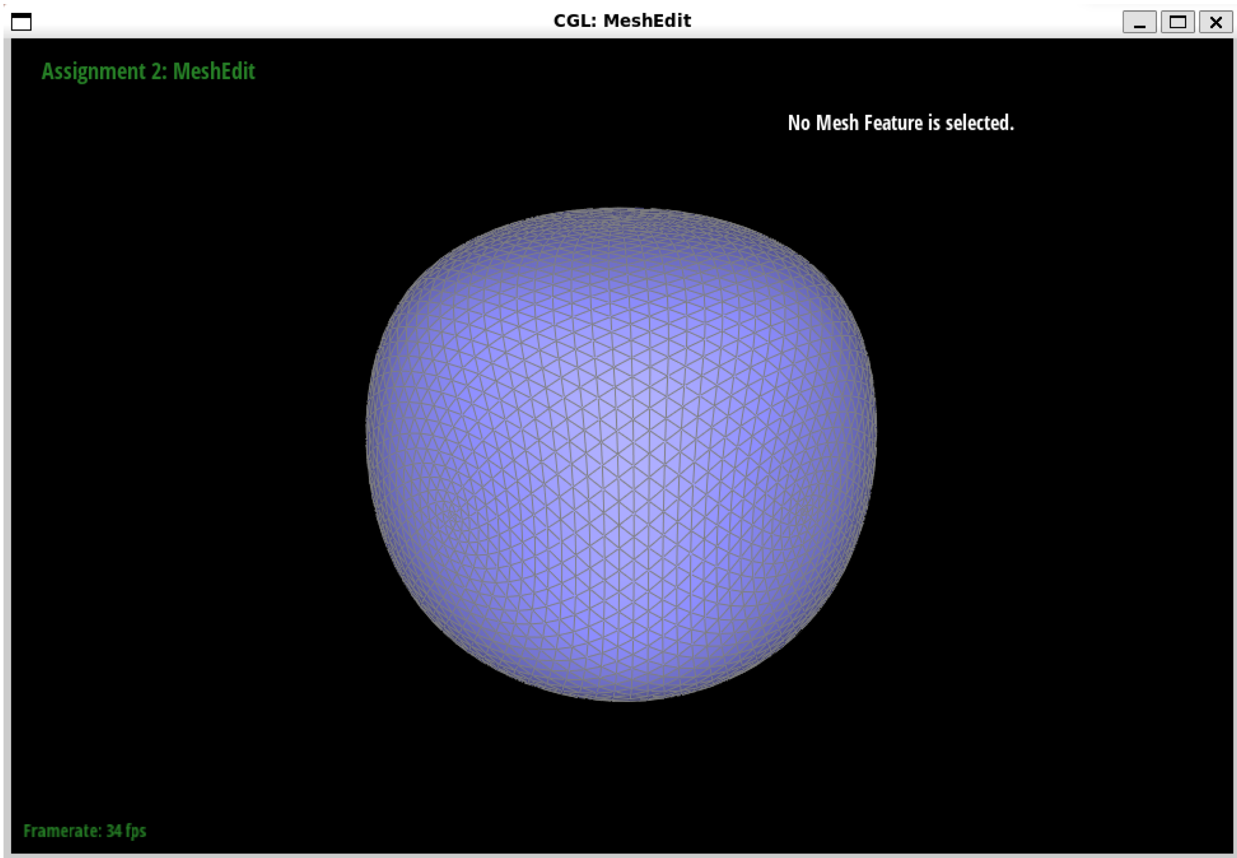

CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 26 fps


CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 0 fps


CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 54 fps


CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 47 fps


CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 44 fps


CGL: MeshEdit
Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 34 fps